PROJECT REQUIREMENT DOCUMENT

Chemical Equipment Parameter Visualizer (Hybrid Web + Desktop App)

1. Project Overview

The Chemical Equipment Parameter Visualizer is a hybrid application that operates both as a Web Application (React.js) and a Desktop Application (PyQt5). Its purpose is to allow users to upload CSV files related to chemical equipment parameters such as Flowrate, Pressure, Temperature, Equipment Name, and Equipment Type. The backend is powered by Django REST Framework, performing data parsing, summary analytics, visualization preprocessing, and returning API responses consumed by both clients. The system also stores the most recent five uploaded datasets and generates PDF reports summarizing the dataset analytics.

2. Technology Stack

Frontend (Web): React.js + Chart.js Frontend (Desktop): PyQt5 + Matplotlib Backend: Python Django + Django REST Framework Data Handling: Pandas Database: SQLite (for this project) Report Generation: ReportLab Version Control: Git & GitHub

3. Database Selection: SQLite

SQLite is chosen for this project due to its simplicity, zero installation requirement, and native integration with Django. It is ideal for lightweight projects such as this internship screening application, where the database stores only the last five uploaded datasets and their summaries. Benefits: • No external installation required • Perfect for small datasets • Excellent Django compatibility • Easy migration to PostgreSQL in the future • Single-file database representation (db.sqlite3)

4. Core Features

1. CSV Upload - Upload CSV from both Web and Desktop versions. - Validate required columns (Flowrate, Pressure, Temperature, etc.). 2. Data Summary Analytics - Total equipment count - Average flowrate, pressure, temperature - Equipment type distribution 3. Data Visualization - React.js uses Chart.js - PyQt5 uses Matplotlib 4. History Management - System stores the latest 5 datasets - Older datasets automatically deleted 5. PDF Report Generation - Summary statistics - Type distribution - Analytics in PDF format 6. Authentication - Basic token-based authentication for API endpoints

5. API Endpoints

• POST /api/auth/login/ • POST /api/upload-csv/ • GET /api/summary// • GET /api/history/ • GET /api/raw// • GET /api/report// • GET /api/download-csv//

6. Application Workflow

1. User logs in and obtains an authentication token. 2. User uploads a CSV file via Web or Desktop app. 3. Django backend receives the file and processes it with Pandas. 4. The backend performs validations and computes a summary. 5. The dataset is stored in SQLite, retaining only the last 5 uploads. 6. Frontend fetches summary & raw data for table/chart display. 7. User can download the dataset summary as a PDF report.

7. Dataset Model (Django ORM)

The system stores dataset metadata and analytics in SQLite using the Django ORM. Below is the schema representation: • filename: Name of uploaded CSV • uploaded_at: Timestamp • csv_file: File path • summary: JSON summary (stored using JSONField) • row_count: Total rows processed • dropped_rows: Invalid or NaN rows removed Only the latest 5 datasets are kept; older ones are deleted automatically.

8. System Advantages

• Hybrid platform: Works on both Web & Desktop • Centralized API ensures consistent results • Lightweight database ensures zero setup • Visual analytics for engineering decision-making • Industry-like dashboard suitable for internship demonstration

End of Document